View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by HKU Scholars Hub

J Intell Manuf (2014) 25:489–503
DOI 10.1007/s10845-012-0701-2

An AIS-based hybrid algorithm for static job shop scheduling problem

Xueni Qiu · Henry Y. K. Lau

Received: 21 May 2012 / Accepted: 18 September 2012 / Published online: 30 September 2012
© The Author(s) 2012. This article is published with open access at Springerlink.com

Abstract A static job shop scheduling problem (JSSP) is a class of JSSP which is a combinatorial optimization problem with the assumption of no disruptions and previously known knowledge about the jobs and machines. A new hybrid algorithm based on artificial immune systems (AIS) and particle swarm optimization (PSO) theory is proposed for this problem with the objective of makespan minimization. AIS is a metaheuristics inspired by the human immune system. Its two theories, namely, clonal selection and immune network theory, are integrated with PSO in this research. The clonal selection theory builds up the framework of the algorithm which consists of selection, cloning, hypermutation, memory cells extraction and receptor editing processes. Immune network theory increases the diversity of antibody set which represents the solution repertoire. To improve the antibody hypermutation process to accelerate the search procedure, a modified version of PSO is inserted. This proposed algorithm is tested on 25 benchmark problems of different sizes. The results demonstrate the effectiveness of the PSO algorithm and the specific memory cells extraction process which is one of the key features of AIS theory. By comparing with other popular approaches reported in existing literatures, this algorithm shows great competitiveness and potential, especially for small size problems in terms of computation time.

Keywords Artificial immune systems (AIS) · Particle swarm optimization (PSO) · Job shop scheduling problem (JSSP) · Clonal selection · Immune network · Memory cells

X. Qiu (✉) · H. Y. K. Lau
Department of Industrial and Manufacturing Systems Engineering, The University of Hong Kong, Hong Kong, People's Republic of China
e-mail: qiuxueni@gmail.com

Introduction

The job shop scheduling problem (JSSP), which is a combinatorial optimization problem, is well-known as a class of NP-hard problem which is unlikely to achieve the global optimal solution in polynomial time. In this work, we assume that most of the information of the system is predefined and there exists no other disturbances, such as machine breakdown, during the whole scheduling process in our system. This problem is called static JSSP. Due to its wide applicability in production and manufacturing industries, and its inherent complexity, it has attracted many researches in the field. As a result, a wide range of approaches and algorithms have been developed over the years.

Generally speaking, these approaches can be divided into two categories, namely, exact methods and approximate methods. The former identifies a precise solution. These methods mainly include enumeration, Lagrangian relaxation, integer programming, dynamic programming, branch and bound (B&B) method. Among these, B&B algorithm is the most popular one. Thus plenty of research has been focused on this method, and considerable advancement has been made (Lageweg et al. 1977; Carlier and Pinson 1989). The algorithm uses a dynamically constructed tree structure to represent all feasible schedules in the search space, and its basic principle is to enumerate all feasible solutions (Brucker et al. 1994). However, as the problem size grows, the exact methods become inefficient and time-consuming because of the computational complexity. They cannot solve large problems within a reasonable time. As such, the research focus has turned into the approximate approaches.

Although approximate methods cannot guarantee the achievement of the global optimal, they are able to find near-optimal solutions for problems of large sizes and even for some complex problems in moderate computing time.

Glover and Greenberg (1989) suggested that the direct tree searching process is unacceptable for complex combinatorial problems, while the heuristics inspired by natural phenomenon and artificial intelligence are more applicable. Therefore, currently there are mainly four types of frequently-used approximation techniques: priority dispatching rules, bottleneck based heuristics, artificial intelligence and local search methods.

Approximation methods applied to static JSSP were first developed based on the priority dispatching rules (PDRs) because of their easy implementation and significant reduction in computational requirement (Baker 1974). In each step, all available operations that can be scheduled are assigned a priority according to the pre-defined rule, and the operation with the highest priority is selected to be executed. However, these methods only consider the current state of the scheduling process rather than the global optimal, and the solution quality is prone to suffer and degrade quickly as the problem size grows.

Subsequently the shifting bottleneck procedure (SBP) was proposed (Adams et al. 1988). It combines schedule construction with iterative improvement and is guided by the one-machine scheduling problem that one-machine relaxation is used to decide the scheduled machine sequence. But this method requires a high level of programming technique.

With the advancement of computer technology, researchers started to work on artificial intelligence and local search methods in the last decade. Artificial Intelligence (AI) methods are developed based on the biological knowledge and principles found in nature to obtain solutions for complex problems. One popular method for static JSSP is the neural network method (Jain and Meeran 1999). It is inspired by the brain structure of simple living entities that information processing is carried out through a huge interconnected network of parallel processing units. Wang and Brunn (1994) presented a review of the application of this method in the scheduling problem, and many researchers (Yahyaoui et al. 2011; Yang et al. 2010; Weckman et al. 2008) have applied it to the static JSSP. Alternatively, the local search method is constructed on the neighborhood structure and the rules which define the way to obtain a new solution from the current one. Its basic idea is to modify the current solutions in terms of the modification method defined by the neighborhood operator, so a new feasible solution is generated which promisingly performs better. Different neighborhood operators or rules generate different meta-heuristic approaches. The most famous ones applied to static JSSP include genetic algorithm (GA) (Pérez et al. 2010), tabu search (TS) (Nowicki and Smutnicki 1996; González et al. 2012; Geyik and Cedimoglu 2004), simulated annealing (SA) (Aydin and Fogarty 2004), and particle swarm optimization (PSO) (Niu et al. 2008), and ant colony optimization (ACO) (Puris et al. 2007). All these methods are well studied and their variations are successfully applied in different domains. As every single technique always exists with some drawbacks, hybridizing is a reasonable way to take strengths and avoid weakness. Hence, the hybrid methods become very popular for the combinatorial optimization problem. For static JSSP, hybrid methods are frequently used, such as hybrid genetic and ant colony heuristics (Girish and Jawahar 2009), hybrid GA-TS (Meeran and Morshed 2011), hybrid PSO with SA (Lin et al. 2010), and hybrid TS-ACO (Eswaramurthy and Tamilarasi 2009). Experiments show that all these hybrid methods perform better than its corresponding single technique because they help each other escape from the local optimal search space and accelerate the convergence rate. A comprehensive and detailed survey of job shop scheduling techniques can be found in (Jain and Meeran 1999).

In terms of the no free lunch (NFL) theory (Wolpert and Macready 1995), no algorithm is always superior to others when compared over all possible issues and every approach is able to exceed at least one subset of certain cases. Thus for the static JSSP, no one method is superior in all situations. Additionally, continuous increase in the problem size increases the complexity and difficulty of the problem for all methods. Therefore, there is still much room for researchers to make improvement and variations to the existing methods and propose new techniques for the problem. Recently, a relatively new theory, artificial immune systems (AIS), has attracted extensive attention owing to its successful applications to many combinatorial optimization problems. Inspired by the human immune system, AIS shows many appealing characteristics, including discrimination of self from non-self, self-learning, long lasting memory, cross reactive response, and strong adaptability to the environment (de Castro and Timmis 2002), which makes it unique from other evolutionary algorithms. It has been successfully applied to the fields of optimization, clustering, pattern recognition, anomaly detection, computer security, machine learning, scheduling, robotics, and control (de Castro and Timmis 2002; Hart and Timmis 2008; Dasgupta et al. 2011; Aydin et al. 2010). For the scheduling problem, AIS has been used in the flow shop scheduling problem (Kahraman et al. 2009), JSSP (Coello et al. 2003), resource constraint project scheduling problem (Mobini et al. 2011), multiprocessor scheduling (Wojtyla et al. 2006), etc. However, for the static JSSP, AIS mechanisms and theories are rarely adopted with only few hybrid AIS approaches (Ge et al. 2008; Zhang and Wu 2010). In this paper, a new hybrid AIS-based algorithm with the meta-heuristics PSO is proposed. Although Ge et al. (2008) have discussed a means to combine AIS with PSO for JSSP, our idea reveals marked differences in the PSO variation and hybrid mode. The proposed hybrid algorithm is studied and compared in terms of solution accuracy and computational efficiency.

The remainder of this paper is organized as follows: "Static job shop scheduling problem" section and "Underlying

theory" section briefly introduce the static JSSP, AIS and PSO theories. In "A hybrid algorithm for static JSSP", the hybrid algorithm is proposed with each step being explained in detail. Based on the benchmark problems, "Experimental analysis" section presents the experimental results and analyzes the algorithm's performance from three aspects. Finally, conclusions are drawn in "Conclusion" section.

Static job shop scheduling problem

The JSSP is a traditional and classical problem. It is defined as follows: given n jobs and m machines in the system with each job consists of m operations, which should be processed by one machine exactly once. At the beginning of the process, all jobs are released and all machines are available. Each machine can manage only one operation at a time, and each job cannot be operated simultaneously by more than one machine. The task is to schedule all the jobs on each machine to achieve the scheduling objective. From a different point of view, the scheduling goal may vary. In this paper, one of the most popular problems—makespan minimization is considered.

Our study focuses on the static JSSP that most of the information about the system is previously known without any unexpected events or machine breakdown during the scheduling process. To simplify the problem, the following assumptions are made: (1) Once an operation has begun on a machine, it cannot be interrupted. (2) The processing time and precedence order of operations for each job are predefined. (3) The jobs can wait between two machines and the intermediate storage is unlimited. (4) Other factors, including machine setup time for two consecutive jobs, transportation time and resource cost, are ignored.

Underlying theory

This paper proposes a new hybrid algorithm based on AIS theory and PSO mechanism for the static JSSP. In the following sections, the two theories are introduced briefly.

Artificial immune systems (AIS)

Artificial immune systems (AIS) is a diverse and maturing artificial intelligence methodology that attempts to bridge the gap between immunology and engineering. It is developed through the application of techniques including mathematical and computational modeling of immunology, abstraction from these models into algorithms, and system design and implementation in the context of engineering (de Castro and Timmis 2002). It has become known as a kind of biologically inspired approach that applies the human immune system metaphors for the creation of novel solutions.

The human immune system is an effective and efficient defense mechanism that protects its host from the invading foreign bodies, called antigens. It behaves as a general and immediate pathogen defense mechanism which combats against a wide variety of foreign invasions without requiring previous exposure to them through the innate immune system, and recognizes previously unknown pathogens (learning) and remembers them for future invasions (memory) through the adaptive immune system (Twycross 2007). The work of the adaptive immune system is performed by two types of lymphocytes, namely, B-cell and T-cell. The former is responsible for the humoral immunity that secretes antibodies binding to antigens by clonal proliferation, while the latter aims at destroying the pathogens directly. These immune cells collaborate in a corporative environment to fight against antigen invasion in the process of recognition, categorization and memorization (de Castro and Timmis 2002).

Inspired by the underlying capability of the human immune system, AIS develops four immunological theories: clonal selection, immune network, negative selection, and dendritic cell algorithm. In this paper, the proposed algorithm integrates clonal selection theory and immune network theory, which encompasses the recognition, selection, maturation, learning and memory processes of the human immune system.

Clonal selection theory

The clonal selection theory presents the fundamental properties of an adaptive immune response to an antigenic stimulus (Timmis 2007). When an antigen invades, the immune system repertoire goes through a selection mechanism that only those antibodies which are capable of recognizing an antigenic stimulus proliferate and differentiate into effective cells. Then these cells suffer somatic hypermutation during reproduction to increase their repertoire diversity and also to become gradually better in their capability of recognizing the selective antigens. During the clonal expansion and mutation process of antibodies, the average antibody affinity increases for the antigen that makes the immune response more effective. This phenomenon is called affinity maturation. In this procedure, the proliferation of antibodies is directly proportional to the affinity of the antigen it binds, and the mutations suffered by the antibodies are inversely proportional to this affinity. That is to say, the higher the antigenic affinity is, the more clones are generated, and the less mutation the antibody occurs. After the cloning and hypermutation process, a percentage of the antibodies with high antigenic affinity are stored as the memory cells to form a large initial specific and efficient antibodies for subsequent re-infections, and some

low antigenic affinity antibodies are assigned to undergo the receptor editing process, in which these less efficient antibodies are replaced by new ones. In terms of the memory cells, the immune system presents the reinforcement learning capability. The receptor editing mechanism develops new antibodies that correspond to new search space, which encourages the algorithm to escape from local optimal.

Immune network theory

Jerne (1974) proposed the immune network theory that presents a novel perspective to important emergent properties of the immune system, such as learning, self-tolerance, and diversity of immune repertoires. It is well used in the machine learning and clustering problems. The premise of this theory is that any antibody molecule could be recognized by the matched antigens and a set of other antibody molecules within the immune system. It is suggested that the immune system is composed of a regulated and stable network of cells that recognizes one another even in the absence of antigens. According to the research in immunology, the recognition of an epitope, which is the decision part of antigen, by a paratope—the recognition part of antibody, results in the reproduction of antibodies with the paratope (stimulation), and the probabilistic reduction of antibodies with the idiotope—the epitope of antibodies (suppression) (de Castro and Timmis 2002). Therefore, the immune system displays a status resulting from interactions among its components and foreign substances.

When translating this feature into an immune-inspired algorithm, it is to compute the affinity amongst all the antibodies and then eliminate those antibodies whose affinity with each other is less than a pre-defined threshold. This action helps to increase diversity of the antibody set.

Particle swarm optimization (PSO)

Particle swarm optimization (PSO), a population based optimization algorithm proposed by Kennedy and Eberhart (1995), is one of the latest evolutionary optimization techniques for continuous optimization problems. It simulates the social interaction and communication in a flock of birds or fishes. In this social group, there is a leader who presents the best performance and guides the movement of the whole swarm. The movement of each particle is directed by the leader and its own knowledge. Thus, the behavior of each particle is a compromise between its individual memory and a collective memory.

In the standard PSO algorithm, each particle in the swarm represents a potential solution. Particle k starts with a random position X_k and a random velocity V_k. During the searching procedure, the particle gains the knowledge about which position P_k^t it has reached presents the best performance, and which position P_g has achieved the best overall performance among all particles. In each iteration t, the behavior of each particle is a compromise among three possible alternatives and its position is updated according to its velocity, shown in Eqs. (1) and (2):

$$V_k^{t+1} = \omega V_k^t + c_1 r_1 \left(P_k^t - X_k^t\right) + c_2 r_2 \left(P_g^t - X_k^t\right) \quad (1)$$

$$X_k^{t+1} = X_k^t + V_k^{t+1} \quad (2)$$

where ω is the inertia weight that controls the impact of the previous velocity on the current velocity. c_1 and c_2 represent the weights of the stochastic acceleration effect when the next position is attracted to the previous best location of the current particle and the whole particle swarm. r_1 and r_2 are two random numbers within the range from 0 to 1. In each iteration, the particle moves according to the reset velocity and position, and the best locations of each particle and the whole swarm are updated for the next generation according the performance of newly generated positions (Xia and Wu 2006). The searching procedure stops when the termination criterion is met.

From the equations, it is seen that the standard PSO is especially suitable for the continuous solution space. Therefore, when applying for the JSSP, a discrete problem, it is necessary to make suitable modifications for PSO to hybridize with AIS. This is discussed in the next section.

A hybrid algorithm for static JSSP

In this section, the hybrid algorithm based on AIS theory and PSO is described for the static JSSP. The flow chart showing the main procedures of the algorithm is shown in Fig. 1. To accelerate the convergence speed of the search algorithm, a neighborhood search mechanism is formulated especially for this problem.

The following paragraphs discuss the key steps in the flow chart in detail.

Antibody generation

In the context of static JSSP, the problem is regarded as the antigen while the antibody corresponds to the scheduling plan. Thus, generating an antibody is equal to creating an initial solution for the problem. The first consideration is how to encode the antibody to represent a feasible schedule. Referring to previous research on the chromosome representation in GA, this algorithm adopts the operation-based representation. Its main advantage is that this representation scheme guarantees that any possible permutation of the random numbers produces a feasible schedule, so the repair mechanism for the infeasible schedule always generated in the antibody mutation process, such as the deadlock schedule

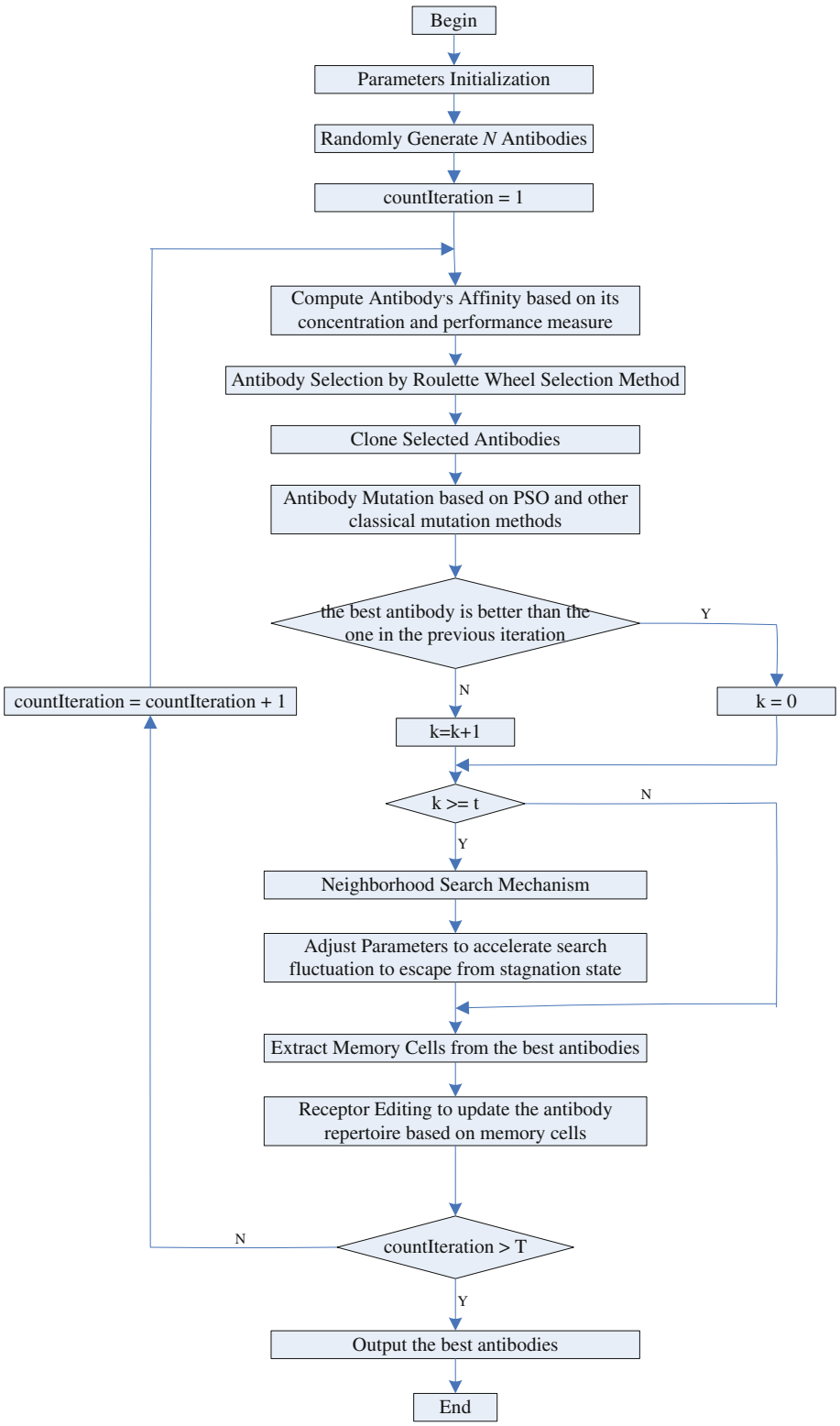

Fig. 1 Flow chart of the hybrid algorithm for static JSSP

that is incompatible with the scheduling constraints and can never be finished, is avoided.

In the operation-based representation approach, the antibody encodes a schedule as a sequence of operations, and defines all the operations for a job as the same number and then interprets it according to the order of occurrence in the given antibody. Thus, an antibody consists of $n \times m$ numbers for an n jobs and m machines problem. Each job index appears m times in the antibody, and each repeated number represents a unique operation of the job. For example, given

an antibody [1, 2, 2, 1, 1, 2], the numbers 1 and 2 represent job 1 and job 2 respectively. In this case, there are three machines and two jobs, and a total of six operations. Each job number is repeated three times. The first integer 2 represents the first operation of job 2, and the second integer 2 represents job 2's second operation. Hence in general, this antibody represents $[O_{11}, O_{21}, O_{22}, O_{12}, O_{13}, O_{23}]$, where O_{ij} stands for the jth operation of job i. However, this representation method produces redundancy in the search space whose size is expanded to $(n \times m)!/(m!)^n$, i.e., different antibodies may represent the same schedule. Therefore, all generated antibodies should be standardized to re-map their relationships with the schedules from many-to-one to one-to-one relationships. This standardization is introduced in "Affinity calculation of antibody" section according to the scheduling plan generated by the antibody.

According to the operation-based representation, the antibodies for the problem with n jobs and m machines are easily produced by ranking $n \times m$ random numbers. Take the 2 jobs and 3 machines problem as an example, we generate a random numerical string [4.5, 0.7, −3.5, 77, −9.2, 6.1] and then rank it in an ascending sequence [4, 3, 2, 6, 1, 5], i.e. the smallest number is −9.2 and we grant the serial number 1 to it; the second one is −3.5 and we grant serial number 2 to it and so on. Finally, divide each serial number by m and round the quotient upwards to the nearest integer. Then the antibody is mapped as [2, 1, 1, 2, 1, 2]. As this random string generation method is prone to create redundant antibody, the newly generated antibody should be compared with existing antibodies such that only a new antibody that is different from the existing ones is accepted.

Affinity calculation of antibody

The antibody's affinity is partly determined by its performance measures, i.e., the makespan of its generated schedule. This is to decode the antibody to the corresponding scheduling plan. By scanning the number of the antibody from left to right, the operation is arranged at the maximum time between the earliest available time of the desired machine and the completion time of the job last preceding operation. This type of schedule is known as semi-active schedule without any excess idle time, but with some "holes". These "holes" can be deleted by shifting some operations to the front without delaying others. This operation is called the "finding and reducing holes procedure", which searches idle time of the working machine and inserts the operation into the idle period without violating the operation precedence constraints. Taking Fig. 2 as an example, the second operation of job 0 working on machine 1 is shifted to fill the holes, and the makespan is reduced by 2 units. This improved schedule is called the active schedule.

According to the generated scheduling plan, the performance measure (makespan) is obtained, i.e., the finish time of the last operation completed in the schedule. Additionally, the antibody can be standardized by reordering the antibody sequence on the basis of the start time of each operation. If the start time is the same, the order is determined by its job index. Thus each schedule corresponds to only one antibody.

In short, the antibody decoding process arranges the active schedule in terms of operation start time, standardizes the antibody to be unique to each schedule, and obtains the makespan value.

Inspired by the immune network theory, it is necessary to consider the population diversity in terms of the antibody similarity with other antibodies to adjust the antibody's affinity. This helps to maintain a high diversity of the antibody set to keep the variety of search space and prevent premature convergence. One method is to delete the antibodies whose affinity amongst all the antibodies, i.e., the reciprocal of concentration, is less than a predefined threshold. However, this operation may leave out some potential good solutions. Therefore, another method which aims to use the antibody's concentration to adjust the antibody's antigenic affinity that is determined by its performance measure is adopted, shown in Eq. (3):

$$\text{Antibody Affinity} = 1/(\text{Makespan} \times \text{Concentration}) \quad (3)$$

The antibody with a higher concentration is assigned a lower affinity with the antigen. The antibody concentration is obtained by averaging the similarity with all other standardized antibodies, which is always measured by the distance

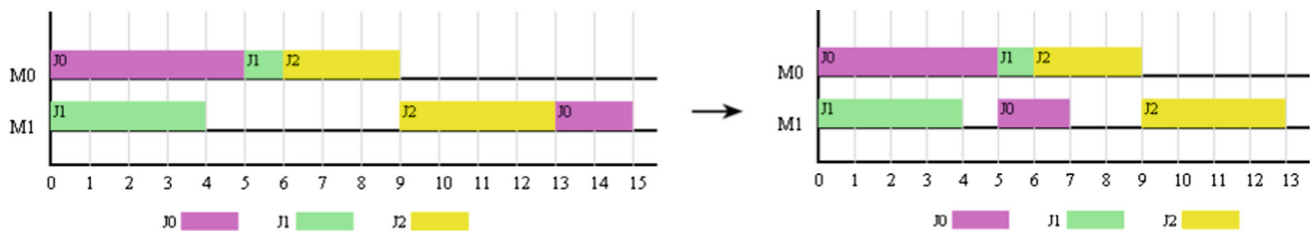

Fig. 2 Converting from semi-active scheduling to active scheduling

between them. As each position of an antibody is represented by a job number without any quantitative meaning and physical association, the commonly adopted distance calculation method—Euclidean distance, cannot measure the difference between two antibodies. Here, the similarity of two antibodies is computed based on the percentage of the same positions between two standardized antibodies. For an n jobs and m machines problem, the concentration of the antibody Ab_i is expressed as SC_i shown in Eq. (4):

$$SC_i = \sum_{j=1}^{N_{Ab}} \sum_{k=1}^{n\times m} Ab_{ij}^k \Big/ [(n \times m) \times N_{Ab}] \quad (i = 1,\ 2 \ldots N_{Ab};\ j = 1,\ 2 \ldots N_{Ab}) \tag{4}$$

where N_{Ab} is the number of antibodies whose affinities are calculated, and the similarity count at the kth locus among the standardized antibody Ab_i and Ab_j is expressed as Ab_{ij}^k shown in Eq. (5):

$$Ab_{ij}^k = \begin{cases} 1 & \text{if the jobs at the k locus of } \mathrm{Ab_i} \text{ and } \mathrm{Ab_j} \text{ are identical} \\ 0 & \text{else} \end{cases} \tag{5}$$

Then the antibody affinity with the antigen is normalized in the range of [0, 1] after all antibodies' affinities with the antigen are computed.

Antibody selection and cloning

The sorted antibodies are selected based on their affinity values to go through the cloning process. The number of antibodies to be selected is $s\% \times N$, where $s\%$ is the selection rate and N is the antibody population size. Compared to the Elitist Selection Approach that only selects the best ones, the Roulette Wheel Selection Method is adopted for its better performance. In this method, the probability of selecting an antibody in the population is directly proportional to its affinity value. Preserving some poorer antibodies with lower affinity values to proliferate is able to advance the diversity of the antibody set and expand the search space. The selected antibodies will then be cloned. The whole clone population is set as the product of three pre-defined parameters, namely, antibody population size N, selection rate $s\%$ and clone amount C. The number of clones of each selected antibody is directly proportional to its affinity value. Hence the number of clones of the selected antibody $i\,(C_i)$ is calculated as:

$$C_i = N \times s\% \times C \times \frac{Aff_i}{\sum_{i=1}^{N\times s\%} Aff_i} \quad (i = 1, 2, \ldots, N \times s\%) \tag{6}$$

where Aff_i is the affinity value of the selected antibody i.

Antibody mutation

In the antibody mutation step, all clones go through the mutation process to convert into some new ones. The mutation rate ($u\%$) is pre-defined. The clonal selection theory suggests that the mutation suffered by the clones is inversely proportional to their antigenic affinity, so the mutation rate of each clone should be further adjusted in a way that is inversely proportionally to its affinity. To keep the mutation procedure in a more diverse manner, three different mutation approaches are applied with different probability. They are the PSO algorithm, point mutation and fragment inverse mutation.

Inspired by the principles of PSO, the proposed algorithm hybridizes with PSO to improve the somatic hypermutation process of the clones to obtain better variations. In the hybrid algorithm, the particle is the antibody. The operators in a classical PSO model, namely, addition and subtraction operators are translated into selection, crossover and mutation processes as shown in Eq. (7).

$$X^k(t+1) = c_1.(P^k(t) - X^k(t)) + c_2.(P_g(t) - X^k(t)) + c_3.[X^k(t)]' \tag{7}$$

Here, the meaning of each variable is the same as for the classical PSO model, while the operations are significantly different. The subtraction operator means the crossover process between the two items. $[X^k(t)]'$ represents the random mutation process of $X^k(t)$. And the addition operator means the selection process among all individuals. Therefore, the newly generated particle $X^k(t+1)$ is set as the best one among $P^k(t) - X^k(t)$, $P_g(t) - X^k(t)$ and $[X^k(t)]'$. The coefficients $c_i\,(i = 1, 2 \text{ and } 3)$ are randomly generated numbers that are controlled by the adjusted mutation rate in the range of [0, 1] for controlling the degree of crossover and mutation process. To simplify the selection process for more efficient runtime, when computing the three items sequentially, once a better item is generated compared with $X^k(t)$, it is assigned as the new particle $X^k(t+1)$ while the others will not be further calculated.

This modified PSO model aims to optimize the mutation process by speeding up the search process in a more efficient way based on the internal acceleration of each particle rather than using random search.

Another two mutation methods are classical ones. One is the point mutation which interchanges two randomly selected jobs of the antibody. The other is the fragment inverse mutation that randomly picks up a continuous fragment of the antibody and reverses the sequence. These two methods are illustrated by Figs. 3 and 4. As the mapping relationship between the non-standardized antibody and the schedule is many-to-one in the operation-based representation, the mutated antibody by exchanging two near jobs or reversing a short fragment of the antibody may yield the

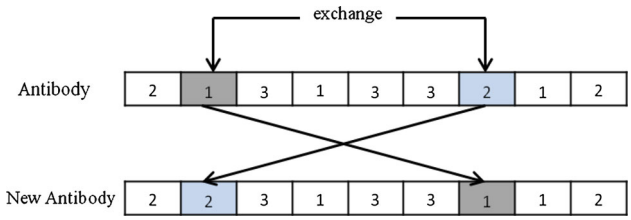

Fig. 3 Point mutation of antibody

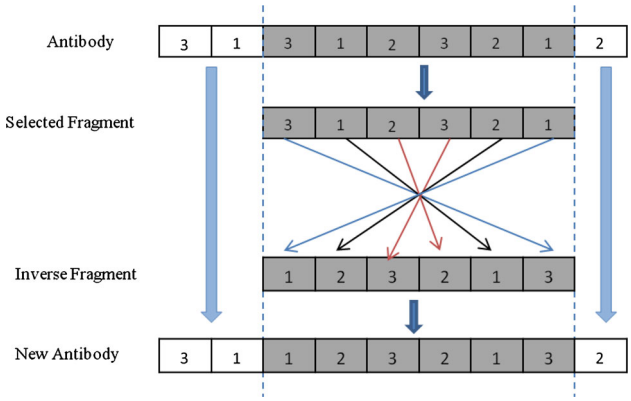

Fig. 4 Fragment inverse mutation of antibody

same schedule as the original one. To minimize such occurrences, the randomly selected two positions in the antibody for exchange should be far away from each other, and the inversed fragment should not be too short. Here, a parameter called "mutation range" (L), is introduced to define the least distance that two exchanged positions should be separated from each other and the least length of the inversed fragment. The larger L is, the more global search effect occurs.

In the mutation process, there are three different mutation approaches. Each clone randomly adopts one of them. Being a novel approach of hybridizing PSO with AIS, it is worthwhile to investigate the effect of PSO on the proposed algorithm. Thus another parameter, called the PSO utilization rate $p\%$, is introduced to define the probability that PSO is applied in the mutation process for a clone. The other two classical mutation methods are used under the same probability, i.e., $(1 - p\%)/2$.

When all the clones finish their mutation processes, there are two choices to reserve the better mutated ones in the repertoire. One way is to pick up a number of the best ones with the highest affinity among all clones and their original antibodies. Another way is to replace each antibody with its own most efficient clone. This approach is better than the former one as it collects only the best one among the mutated clones from each antibody to keep the diversity of the antibody set. So the second method is adopted.

Neighborhood search mechanism

When decoding the antibody to obtain the scheduling plan, the active schedule is adopted instead of the semi-active one by "finding and reducing holes procedure". Here, another neighborhood search mechanism is introduced to find better neighborhood solutions for the active schedule.

This method defines a neighborhood solution based on the concept of blocks. A block is defined as the consecutive operations processed on the same machine in the critical path, that is, the longest continuous path in the corresponding Gantt Chart of the schedule. It has three characteristics: the operations in the block belong to the critical path; each block contains the operations processed on the same machine; the operations in two consecutive blocks are processed on different machines. By moving the operations nearer to the border line of blocks on the critical path, a better neighbor solution may be generated. This is done by swapping the last two operations in the first block, or the first two operations in the last block, or the first two or the last two operations for other blocks, because swapping other adjacent operations in blocks has been demonstrated to be ineffective (Nowicki and Smutnicki 1996). Taking Figs. 5 and 6 as an example, the operations in the critical path are labeled by the red "$\surd$". There are four blocks in Fig. 5, and three blocks in Fig. 6. Only the third block in Fig. 5 and the first and second block in Fig. 6 are applicable to the neighborhood search mechanism. Exchange two operations in the third block of Fig. 5, i.e., the third operation of job 1 and the third operation of job 2. A new schedule is generated as shown in Fig. 6. The makespan is as a result shortened by three units.

As this neighborhood search mechanism is computationally complex and relatively time-consuming, it is applied only when the algorithm is trapped in a local optimum. Therefore, in this algorithm, this method and the following parameter adjustment step are executed only when the best performance measure, i.e., minimal makespan, in the antibody set has not improved in consecutive t iterations.

Parameters adjustment

A large number of parameters including antibody population size N, clone amount C, selection rate $s\%$, mutation rate $u\%$, mutation range L, and replacement rate $r\%$ are used in the next step. As mentioned previously, when the solution is trapped in a local optimum, the parameters are modified to increase the search fluctuation so as to help the algorithm escape from the stagnation stage and widen the search space globally. The parameter adjustment algorithm therefore increases the values of $u\%$, L, C and $r\%$ by 10 %.

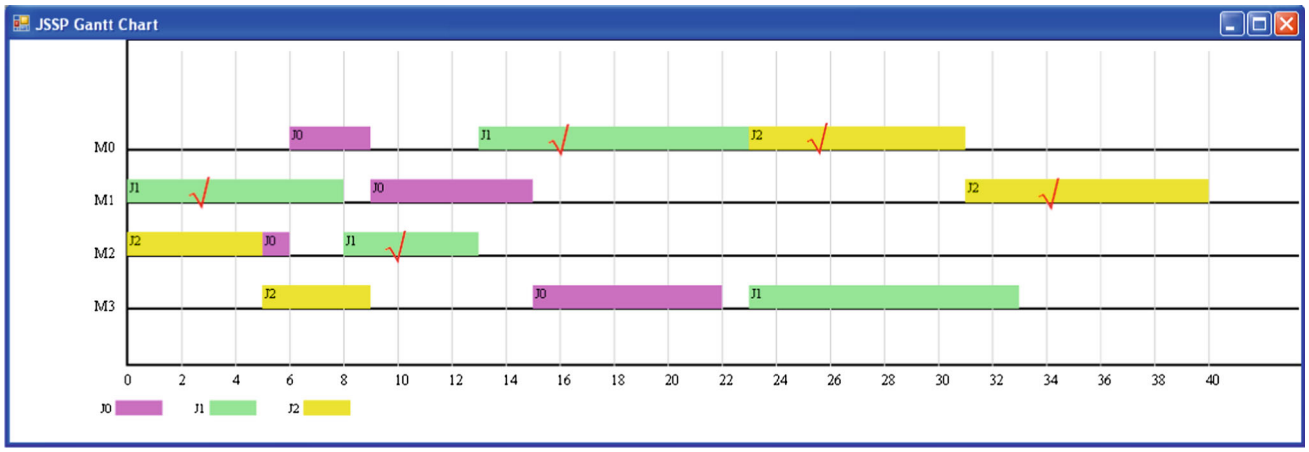

Fig. 5 Gantt chart of the original schedule

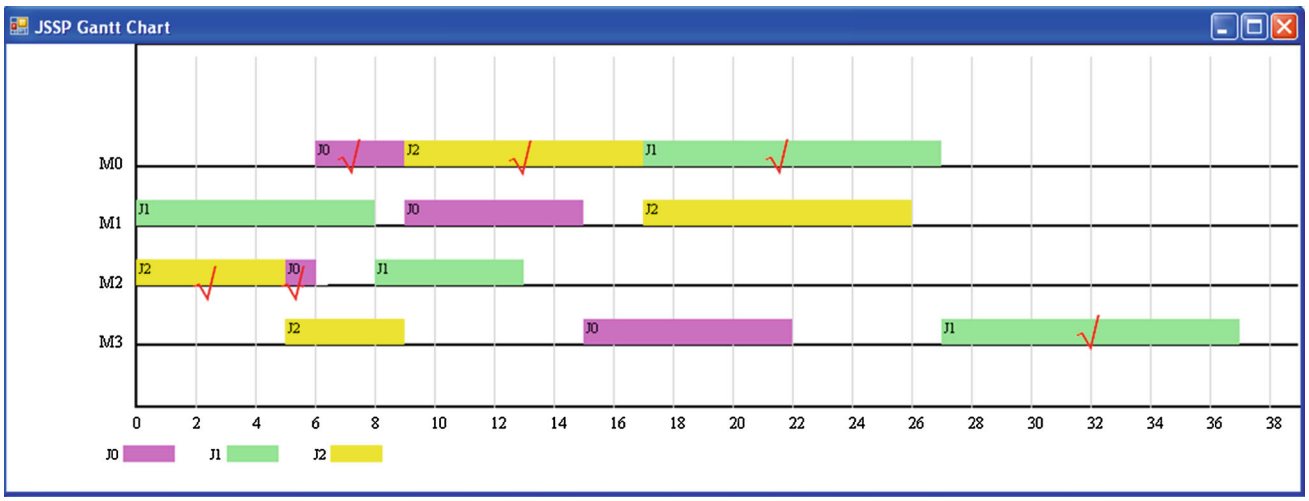

Fig. 6 Gantt chart of the schedule that is improved by neighborhood search mechanism

Memory cells extraction

The antibodies with the highest affinity are assigned to differentiate into memory cells for future re-infections. Rather than "starting from scratch" every time, the memory cells help both the speed and accuracy of the solution process as the immune response becomes increasingly stronger after each infection. Thus the system continuously learns from the direct interaction with the environment. As such, the memory cells extraction process is adopted to advance the antibody (scheduling plan) to accelerate the convergence in the proposed method.

In each iteration, there always exists more than one best antibody (solution) with the same highest affinity value (the predominant objective value). During the search procedure, all these best antibodies are selected for memory cells extraction. The characteristic of the best antibodies to be picked up as the memory cells is decided by the problem to be solved. The static JSSP's solution is to identify the start operating time of each job on each machine. But the start time of each operation is changeable because it always varies for all antibodies even with the same objective value. In addition, the operation start time cannot take a fixed value as it is controlled and constrained by the job sequence and processing time constraints. A small fluctuation of one operation start time may result in a totally different scheduling plan with the same performance. As such, each operation start time is not suitable to be the memory cells. Therefore, another factor is considered. From each solution, we also get the job sequence for each machine and this information provides important information for the scheduling plan. Suppose a schedule is efficient, its corresponding job sequences on each machine are also good. For a certain machine, different best solutions always have the same job sequence. Thus this sequence can be considered good for this machine, and be used in other solutions to improve their quality. Moreover, unlike the operation start

time, any jobs permutation on a machine is feasible because it is not restricted by the problem's constraints. Therefore, the sequence of processing jobs on each machine can be a good candidate of memory cells.

At the end of each iteration, all the best antibodies are collected. Their corresponding job sequences on each machine are obtained. Suppose there are m machines in the problem, the memory cells are therefore divided into m groups. Each group represents a machine and consists of several memory cells. Each memory cell represents a job sequence on this machine, and it is assigned an affinity value represented by the frequency that it appears in all the best antibodies. Then these memory cells are used to generate new antibodies in the next step.

Receptor editing

The receptor editing process aims to generate new antibodies to escape from the unsatisfactory local optimum. In this step, $r\%$ (replacement rate) less efficient antibodies which gains the lowest affinity are replaced by the new ones generated from the memory cells.

The number of the selected memory cells should be in the range of $[0, m]$, because there are only m memory cell groups and the cells in each group are incompatible. It is known that the memory cells become more mature in later iterations. Therefore, the number of memory cells picked up from the memory cell set is proportional to the iteration number. Hence, the later iterations will extract more memory cells, i.e., more machines' job sequence will be pre-defined. In the last iteration, all machines' job sequences are determined by the memory cells, while in the first iteration, relatively few machines' job sequences (m/T) are defined, and others should be generated randomly that are complied with the job sequence constraints.

In the memory cells selection step, only one memory cell is picked from each memory cell group. In each memory cell group, the selection rate of each memory cell is proportional to its affinity value. The higher affinity a memory cell has, the higher probability this memory cell is to be selected. Then the selected memory cells are combined to generate a new antibody. In some cases, some memory cells from different groups are incompatible because of the logical paradox in the job sequence. In such cases, this combination is ignored, and the memory cells are re-selected.

Experimental analysis

The proposed algorithm is evaluated on a number of well-known benchmark problems in the OR-library (Beasley 1990), and compared with other approaches to show its performance. 24 instances in seven groups which differ in the problem sizes (n jobs $\times m$ machines) are selected from the OR-library. They are FT06 (6×6), LA01 (10×5), FT10 and ORB01 $\sim$ ORB10 (10×10), LA06 (15×5), LA21 (15×10), LA11 $\sim$ LA14 and FT20 (20×5), LA26 $\sim$ LA27 (20×10), and ABZ8 $\sim$ ABZ9 (20×15). The parameters are set differently based on the problem size as follows: $N = 2 \times m \times n$, $C = 20$, s% $= 0.7$, u% $= 0.6$, $L = m$, $r\% = 0.2$, T $= m \times n$, $t = 0.1$ T.

As the novelty of this algorithm lies in the integration of AIS and PSO, it is necessary to perform sensitivity analysis for the parameter—PSO utilization rate $p\%$, to show the effect of PSO in the hybrid algorithm. Additionally, one of the distinct features of AIS, memory cells which are exacted from previous outstanding antibodies for the next immune response, is investigated. The experiments are designed to consider the following three perspectives: firstly, to perform sensitivity analysis on $p\%$; secondly, to evaluate the hybrid algorithm and compare it with other similar approaches; and thirdly, to demonstrate the effectiveness of the memory cells mechanism.

Sensitivity analysis on parameter $p\%$

This experiment aims to demonstrate the usefulness and effectiveness of PSO in the proposed hybrid algorithm. If the hybrid algorithm shows the best performance when $p\%$ is set as zero, PSO will be regarded as redundant. Here, six problems with four different sizes, namely, ORB02, ORB04 and ORB06 (10×10), LA06 (15×5), LA21 (15×10), and FT20 (20×5) are used to investigate the impact of PSO on the hybrid algorithm. The PSO utilization rate $p\%$ is tested on 11 cases. They are 0, 0.1, 0.2, 0.3, …, 1.0. For each case, these six problems are solved by the algorithm for 50 times repeatedly and independently with the same parameters. Figure 7 shows the relationship between $p\%$ and the average makespan value. The curve is in a "V" shape showing the algorithm performs well when $p\%$ is in the range of [0.4, 0.7]. When $p\%$ is less than 0.4, the average objective value decreases as $p\%$ grows. This exemplifies that the PSO variation has a positive influence to the mutation process in helping the antibodies move to a more favorable search space. On the other hand, when $p\%$ is greater than 0.7, the average makespan value increases rapidly with increasing $p\%$. This illustrates the negative impact on the performance when PSO is excessively used in the mutation process. In summary, PSO helps the proposed algorithm accelerate the hypermutation process of the clones, though the mutation process should be varied and be controlled by different mutation methods, otherwise, the algorithm is prone to converge to some local optimum in the search space. Thus an optimal value of $p\%$ should be set in the range of [0.4, 0.7].

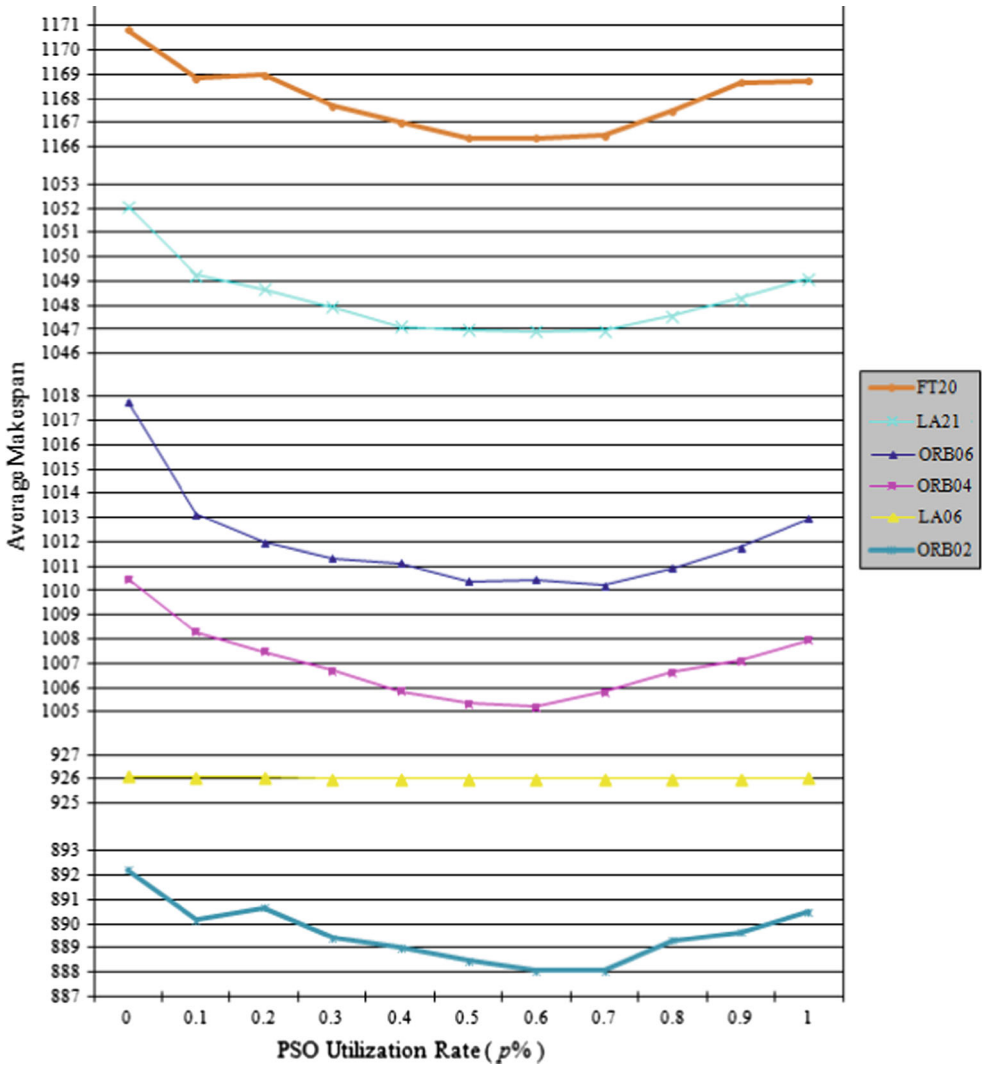

Fig. 7 The relationship between $p\%$ and the average makespan

Performance of the hybrid algorithm

The proposed algorithm is evaluated by solving the benchmark problems. Initially, the parameter $p\%$ is set to 60 %. Depending on the performance of the algorithm, the value of $p\%$ is adjusted within the range of [0.4, 0.7]. For each case, the algorithm is executed 50 times independently to compute the average value (Avg.), standard deviation (SD) and average relative error (Avg. RE). The results are summarized in Table 1. In our experiments, the relative error is defined as the percentage deviation from the best known solution, as shown in Eq. (8).

$$\text{Relative Error } (\%) = (\text{heuristic solution} - \text{best solution})/\text{best solution} \times 100 \quad (8)$$

The performance of the algorithm is compared with the greedy randomized adaptive search procedure (GRASP), modified genetic algorithm (GA), best-so-far artificial bee colony (best-so-far ABC), multi-modal immune algorithm, and other AIS based or hybrid algorithms discussed in the references (Binato et al. 2002; Wang and Zheng 2002; Luh and Chueh 2009; Chandrasekaran et al. 2006; Ge et al. 2008; Coello et al. 2003). From Table 1 and related literatures from those references, the algorithm performs competitively in all cases as it achieves the global optimal solution even for large-size problems. In terms of the computation time, it is in the acceptable range. With this parameter setting, the average computation time for problem sizes of 6×6, 10×5, 10×10, 15×5, 15×10, 20×5, 20×10, 20×15 require about 20 s, 30 s, 5 min, 2.5 min, 20 min, 4 min, 52 and 55 min respectively to run on a Intel Core 2 Quad Q9400 (2.66 GHz, 2 GB RAM) PC under Microsoft Windows XP Professional Operating

Table 1 Experimental results of solving different problems instances

Problem	Size	Best known	Optimal	Avg.	SD	Avg. RE (%)
FT06	6 × 6	55	55	55	0.0	0.0
LA01	10 × 5	666	666	666	0.0	0.0
FT10	10 × 10	930	930	930	0.0	0.0
ORB01	10 × 10	1,059	1,059	1,059	0.0	0.0
ORB02	10 × 10	888	888	888.06	0.23990	0.00676
ORB03	10 × 10	1,005	1,005	1,005	0.0	0.0
ORB04	10 × 10	1,005	1,005	1,005.22	0.88733	0.0219
ORB05	10 × 10	887	887	887	0.0	0.0
ORB06	10 × 10	1,010	1,010	1,010.22	0.76372	0.02178
ORB07	10 × 10	397	397	397	0.0	0.0
ORB08	10 × 10	899	899	899	0.0	0.0
ORB09	10 × 10	934	934	934	0.0	0.0
ORB10	10 × 10	944	944	944	0.0	0.0
LA06	15 × 5	926	926	926	0.0	0.0
LA21	15 × 10	1,046	1,046	1,046.92	2.33728	0.0879
FT20	20 × 5	1,165	1,165	1,166.38	3.26946	0.1185
LA11	20 × 5	1,222	1,222	1,222	0.0	0.0
LA12	20 × 5	1,039	1,039	1,039	0.0	0.0
LA13	20 × 5	1,150	1,150	1,150	0.0	0.0
LA14	20 × 5	1,292	1,292	1,292	0.0	0.0
LA26	20 × 10	1,218	1,218	1,218	0.0	0.0
LA27	20 × 10	1,235	1,239	1,245.98	8.01908	0.88907
ABZ8	20 × 15	670	672	677.12	9.16914	1.14007
ABZ9	20 × 15	691	693	694.68	3.81030	0.49973

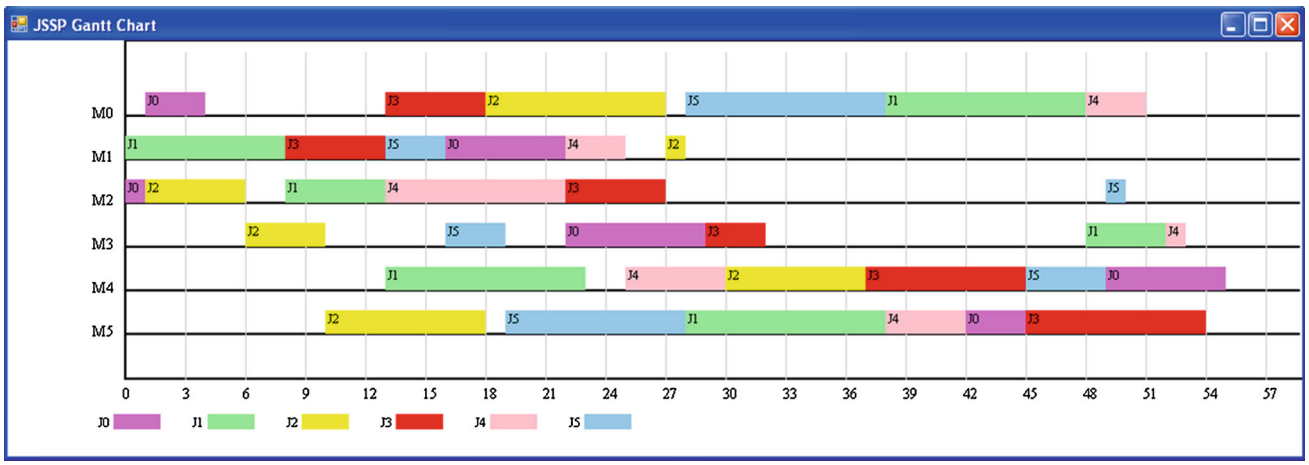

Fig. 8 Gantt chart showing the best scheduling for FT06 (1)

System. To reduce the computation time without deteriorating the performance, it is possible to decrease the maximal iteration number T as the algorithm always converges to the optimal solution in less than T iterations for some instances, especially for small size problems. For example, when solving the cases FT06 (6 × 6), LA01 (10 × 5), LA06 (15 × 5) and LA11 ∼ LA14 (20 × 5), the algorithm obtains the optimal solution in the first iteration. Thus the computation time for these problems is actually about 0.56, 0.6, 2 and 2.4 s to achieve the best solutions.

Moreover, the proposed algorithm is able to obtain multiple optimal solutions with the same objective value as in each

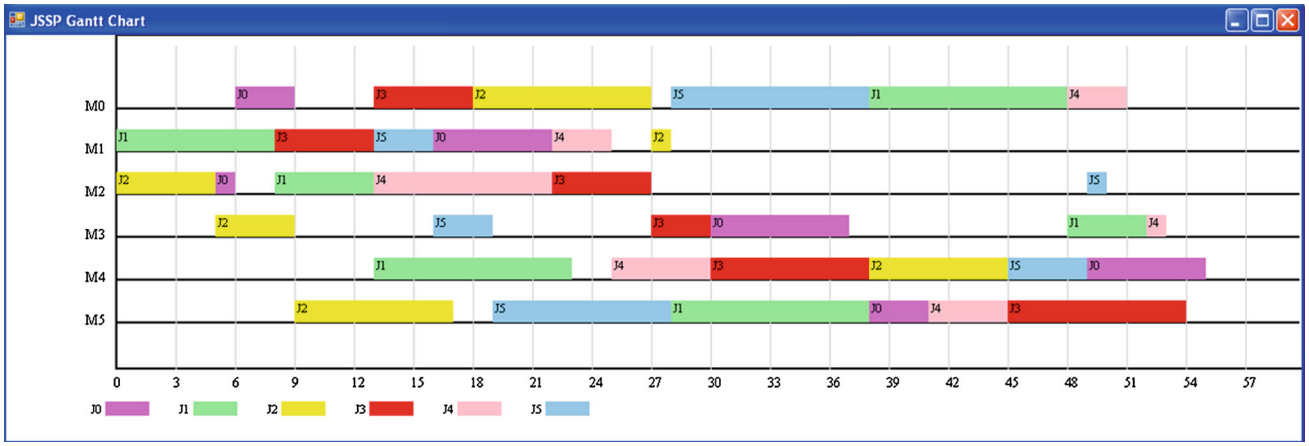

Fig. 9 Gantt chart showing the best scheduling for FT06 (2)

Table 2 Comparison of the two approaches

Problem	Size	Best known	Method	Optimal	Avg.	SD	Avg. RE (%)
ORB02	10 × 10	888	Proposed algorithm	888	888.06	0.23990	0.00676
			Modified version	888	888.24	0.1263	0.0096
ORB04	10 × 10	1005	Proposed algorithm	1,005	1,005.22	0.88733	0.0219
			Modified version	1005	1005.30	1.0274	0.0311
ORB06	10 × 10	1010	Proposed algorithm	1,010	1,010.22	0.76372	0.02178
			Modified version	1010	1010.40	0.9215	0.0248
LA21	15 × 10	1046	Proposed algorithm	1,046	1,046.92	2.33728	0.0879
			Modified version	1046	1047.46	2.68184	0.13958
FT20	20 × 5	1165	Proposed algorithm	1,165	1,166.38	3.26946	0.1185
			Modified version	1165	1167.04	3.48150	0.17511
LA26	20 × 10	1218	Proposed algorithm	1,218	1,218	0.0	0.0
			Modified version	1218	1218.04	0.282843	0.00328
A27	20 × 10	1235	Proposed algorithm	1,239	1,245.98	8.01908	0.88907
			Modified version	1250	1253.16	3.649769	1.47045
ABZ8	20 × 15	670	Proposed algorithm	672	677.12	9.16914	1.14007
			Modified version	674	678.92	9.6459	1.2194
ABZ9	20 × 15	691	Proposed algorithm	693	694.68	3.81030	0.49973
			Modified version	693	695.02	4.0071	0.5843

iteration the antibodies in the repertoire can be different. This provides alternative scheduling plans for the decision maker who can take into account other practical factors and adopt the most suitable and appropriate optimal setting. For example, the algorithm obtains 22 different scheduling plans for the problem FT06 (6 × 6) with the same minimal makespan value of 55 where two of these are shown in Figs. 8 and 9. To improve the robustness of the scheduling system in real situations, it is suggested to increase the adaptability and flexibility of each operation by maximizing the slack time of each operation. This helps the scheduling process reduce the negative effect caused by some unexpected events, such as the fluctuation of each operation's processing time and machine breakdown. Thus the first solution is selected from Figs. 8 and 9.

Effectiveness of memory cells

To show the effectiveness of introducing the memory cells, an experiment based on the benchmark problems is designed to compare the proposed algorithm with its modified version where memory cells extraction step is taken out. The results are shown in Table 2, where only the cases with positive average relative error are presented. It is obvious that the

Table 3 Average iteration number to achieve the optimal solution

Problem	Size	Total iteration no. (T)	Average iteration number to reach the optimal solution	
			Proposed algorithm	Modified version (without memory cells extraction)
FT06	6×6	36	1	1
LA01	10×5	50	1	1.04
FT10	10×10	100	16.42	44.78
ORB01	10×10	100	13.80	38.06
ORB03	10×10	100	48.04	74.56
ORB05	10×10	100	22.70	44.16
ORB07	10×10	100	11.78	23.90
ORB08	10×10	100	28.54	57.40
ORB09	10×10	100	19.66	51.00
ORB10	10×10	100	15.28	43.36
LA06	15×5	75	1	1.08
LA11	20×5	100	1	1.02
LA12	20×5	100	1	1.10
LA13	20×5	100	1	1.04
LA14	20×5	100	1	1.08

introduction of memory cells increases the chance to obtain the optimal solution.

Additionally, for the cases where both approaches achieve the optimal solution in all the runs, it is necessary to compare the number of iterations when the approach first achieves the optimal solution. The experiments results, as shown in Table 3, illustrate that the algorithm with memory cells introduced obtains the best schedule in fewer iterations. Therefore, it is concluded that the use of memory cells accelerates the convergence rate of the algorithm and reduces the computation time.

Conclusions

This paper proposes a new hybrid algorithm based on the clonal selection, immune network, and PSO theories for solving static JSSP. The algorithm simulates the clonal selection process of antibodies with improvement of the mutation process by PSO. To demonstrate its feasibility and efficiency, experiments are designed with the benchmark problems under three perspectives. Firstly, PSO is demonstrated to make a positive impact on the mutation process to a certain degree as excessive use deteriorates the objective value. Secondly, the algorithm is tested on 25 benchmark problems and compared with other popular approaches. The results demonstrate the competitiveness of the proposed algorithm where multiple optimal solutions are obtained within reasonable computation time, especially for some small size problems in which the algorithm achieves the optimal solution in less than 3 s. Thirdly, one of the key features of AIS—immune cells extraction is demonstrated to have the ability to accelerate the convergence.

As AIS show good learning and memory capabilities and strong adaptability to the dynamic environment, it is possible to adapt other immune theories that are associated with self-regulation and control of dynamic situations, such as dendritic cell algorithm and idiotypic network, to deal with dynamic JSSP where unexpected disturbances are considered in the scheduling process. We find that this research direction is fruitful in practice in our current research.

Acknowledgments The authors gratefully acknowledge the financial support from the Research Grant Council of the HKSAR Government, P. R. China.

Open Access This article is distributed under the terms of the Creative Commons Attribution License which permits any use, distribution, and reproduction in any medium, provided the original author(s) and the source are credited.

References

Adams, J., Balas, E., Zawack, D. (1988). The shifting bottleneck procedure for job shop scheduling. *Management Science, 34*(3), 391–401.

Aydin, I., Karakose, M., & Akin, E. (2010). An adaptive artificial immune system for fault classification. *Journal of Intelligent Manufacturing, 23*(5), 1489–1499.

Aydin, M. E., & Fogarty, T. C. (2004). A simulated annealing algorithm for multi-agent systems: A job-shop scheduling application. *Journal of Intelligent Manufacturing, 15*(6), 805–814.

Baker, K. (1974). *Introduction to sequencing and scheduling*. New York: Wiley.

Beasley, J. (1990). OR-library: Distributing test problems by electronic mail. *The Journal of the Operational Research Society, 41*(11), 1069–1072.

Binato, S., Hery, W. J., Loewenstern, D. M., & Resende, M. G. C. (2002). A GRASP for job shop scheduling. *Essays and Surveys in Metaheuristics, 15*, 59–79.

Brucker, P., Jurisch, B., & Sievers, B. (1994). A branch-and-bound algorithm for the job-shop scheduling problem. *Discrete Applied Mathematics, 49*(1–3), 107–127.

Carlier, J., & Pinson, E. (1989). An algorithm for solving the job-shop problem. *Management Science, 35*(2), 164–176.

Chandrasekaran, M., Asokan, P., Kumanan, S., Balamurugan, T., & Nickolas, S. (2006). Solving job shop scheduling problems using artificial immune system. *International Journal of Advanced Manufacturing Technology, 31*(5–6), 580–593.

Coello, C. A. C., Rivera, D. C., & Cortes, N. C. (2003). Use of an artificial immune system for job shop scheduling. *Proceedings of the Second International Conference of Artificial Immune Systems, 2787*, 1–10.

Dasgupta, D., Yu, S., & Nino, F. (2011). Recent advances in artificial immune systems: Models and applications. *Applied Soft Computing, 11*(2), 1574–1587.

de Castro, L. N., & Timmis, J. (2002). *Artificial immune systems: A new computational intelligence approach*. New York: Springer.

Eswaramurthy, V. P., & Tamilarasi, A. (2009). Hybridizing tabu search with ant colony optimization for solving job shop scheduling problems. *International Journal of Advanced Manufacturing Technology, 40*(9–10), 1004–1015.

Ge, H. W., Sun, L., Liang, Y. C., & Qian, F. (2008). An effective PSO and AIS-based hybrid intelligent algorithm for job-shop scheduling. *IEEE Transactions on Systems Man and Cybernetics Part a-Systems and Humans, 38*(2), 358–368.

Geyik, F., & Cedimoglu, I. H. (2004). The strategies and parameters of tabu search for job-shop scheduling. *Journal of Intelligent Manufacturing, 15*(4), 439–448.

Girish, B. S., & Jawahar, N. (2009). Scheduling job shop associated with multiple routings with genetic and ant colony heuristics. *International Journal of Production Research, 47*(14), 3891–3917.

Glover, F., & Greenberg, H. J. (1989). New approaches for heuristic-search—a bilateral linkage with artificial-intelligence. *European Journal of Operational Research, 39*(2), 119–130.

González, M. A., Vela, C. R., González-Rodríguez, I., & Varela, R. (2012). Lateness minimization with Tabu search for job shop scheduling problem with sequence dependent setup times. *Journal of Intelligent Manufacturing*. doi:10.1007/s10845-011-0622-5.

Hart, E., & Timmis, J. (2008). Application areas of AIS: The past, the present and the future. *Applied Soft Computing, 8*(1), 191–201.

Jain, A., & Meeran, S. (1999). A state-of-the-art review of job-shop scheduling techniques. *European Journal of Operations Research, 113*(2), 390–434.

Jerne, N. K. (1974). Towards a network theory of the immune system. *Ann Immunol, 125*(1–2), 373–389.

Kahraman, C., Engin, O., & Yilmaz, M. K. (2009). A new artificial immune system algorithm for multiobjective fuzzy flow shop. *International Journal of Computational Intelligence Systems, 2*(3), 236–247.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *IEEE International Conference on Neural Network, 4*, 1942–1948.

Lageweg, B. J., Lenstra, J. K., & Rinnooy Kan, A. H. G. (1977). Job-shop scheduling by implicit enumeration. *Management Science, 24*(4), 441–450.

Lin, T. L., Horng, S. J., Kao, T. W., Chen, Y. H., Run, R. S., & Chen, R. J., et al. (2010). An efficient job-shop scheduling algorithm based on particle swarm optimization. *Expert Systems with Applications, 37*(3), 2629–2636.

Luh, G. C., & Chueh, C. H. (2009). A multi-modal immune algorithm for the job-shop scheduling problem. *Information Sciences, 179*(10), 1516–1532.

Meeran, S., & Morshed, M. (2011). *Journal of Intelligent Manufacturing, 23*(4), 1063–1078.

Mobini, M., Mobini, Z., & Rabbani, M. (2011). An artificial immune algorithm for the project scheduling problem under resource constraints. *Applied Soft Computing, 11*(2), 1975–1982.

Niu, Q., Jiao, B., & Gu, X. S. (2008). Particle swarm optimization combined with genetic operators for job shop scheduling problem with fuzzy processing time. *Applied Mathematics and Computation, 205*(1), 148–158.

Nowicki, E., & Smutnicki, C. (1996). A fast taboo search algorithm for the job shop problem. *Management Science, 42*(6), 797–813.

Pérez, E., Posada, M., & Herrera, F. (2010). Analysis of new niching genetic algorithms for finding multiple solutions in the job shop scheduling. *Journal of Intelligent Manufacturing, 23*(3), 341–356.

Puris, A., Bello, R., Trujillo, Y., Nowe, A., & Martinez, Y. (2007). Two-stage ACO to solve the job shop scheduling problem. In *Proceedings of the congress on pattern recognition 12th Iberoamerican conference on progress in pattern recognition, image analysis and applications* (Vol. 4756, pp. 447–456).

Timmis, J. (2007). Artificial immune systems: Today and tomorrow. *Natural Computing, 6*(1), 1–18.

Twycross, J. (2007). *Integrated innate and adaptive artificial immune systems applied to process anomaly detection.* PhD thesis, The University of Nottingham, UK.

Wang, L., & Zheng, D. Z. (2002). A modified genetic algorithm for job shop scheduling. *International Journal of Advanced Manufacturing Technology, 20*(1), 72–76.

Wang, W., & Brunn, P. (1994). Production scheduling and neural networks. In *Operation Research Proceedings*, 173–178.

Weckman, G. R., Ganduri, C. V., & Koonce, D. A. (2008). A neural network job-shop scheduler. *Journal of Intelligent Manufacturing, 19*(2), 191–201.

Wojtyla, G., Rzadca, K., & Seredynski, F. (2006). Artificial immune systems applied to multiprocessor scheduling. *Parallel Processing and Applied Mathematics, 3911*, 904–911.

Wolpert, D. H., & Macready, W. G. (1995). No free-lunch theorems for search. Working paper 95-02-010, Santa Fe Institute.

Xia, W. J., & Wu, Z. M. (2006). A hybrid particle swarm optimization approach for the job-shop scheduling problem. *International Journal of Advanced Manufacturing Technology, 29*(3–4), 360–366.

Yahyaoui, A., Fnaiech, N., & Fnaiech, F. (2011). A suitable initialization procedure for speeding a neural network job-shop scheduling. *IEEE Transactions on Industrial Electronics, 58*(3), 1052–1060.

Yang, S. X., Wang, D. W., Chai, T. Y., & Kendall, G. (2010). An improved constraint satisfaction adaptive neural network for job-shop scheduling. *Journal of Scheduling, 13*(1), 17–38.

Zhang, R., & Wu, C. (2010). A hybrid immune simulated annealing algorithm for the job shop scheduling problem. *Applied Soft Computing, 10*(1), 79–89.